애플리케이션 통합을 위한 워크플로우 관리 시스템

윤 석 현[†] · 오 명 은[†] · 한 상 용[††]

요 약

현대 사회의 급변하는 기업환경에 대처하기 위해서 기업들은 기존의 애플리케이션 사이에 중간 매개체인 미들웨어를 설치하여 애플리케이션 통합을 이룰 수 있는 EAI를 도입하고 있다. 그러나 이러한 미들웨어 기반의 EAI는 해당 미들웨어에 종속적인 부가 코딩을 필요로 한다는 문제점이 있다. 본 논문에서는 스테이트 차트와 XML 기술을 이용하여 다양한 미들웨어 시스템 상의 애플리케이션을 효율적으로 통합할 수 있는 이중(duplicate) 어댑터 기반의 미들웨어 독립적 워크플로우 관리 시스템을 제안한다.

Workflow Management System for Application Integration

SeokHyun Yoon[†] · Myeongeun Oh[†] · Sangyong Han[††]

ABSTRACT

To cope with rapidly changing enterprise environment, many business entities introduce EAI, which integrates applications by installing middleware. But existing EAI has a serious problem that is to import middleware-dependnt coding method to integrate systems, using different middlewares. In this paper, we suggest middleware independent 'Workflow Management System' which efficiently integrates applications of various middleware systems with 'Duplicate EAI Adapter,' 'state chart' and 'XML' technology.

키워드 : 워크플로우(Workflow), EAI, WFMS(Workflow management System)

1. 서 론

지난 수 년 동안 경영 환경은 기업에게 있어 보다 핵심적이고 빠른 시간에 대응할 수 있는 방안을 중심으로 급변해 왔다. 대부분의 기업들은 핵심 요소 기술 개발에 주력하고, 나머지 요소들에 대해서는 외주를 통해 해결하는 방식을 취하고 있다. 이러한 이유로 기업 환경에서는 특정 작업을 위해서 하나의 새로운 애플리케이션을 개발하여 일괄처리를 하기보다는 기존의 애플리케이션에 외부의 애플리케이션을 도입하여 그들을 유기적으로 결합하는 형태로 원하는 기능을 수행하는 사례가 늘어나고 있다. 이와 같이 기업 전체를 통해서 상호 연관된 모든 애플리케이션을 유기적으로 연동시켜 필요한 정보를 통합하고 관리, 사용할 수 있게 하는 EAI(Enterprise Application Integration)와 여러 애플리케이션들 사이의 협력 작업을 정해진 절차에 의해서 수행되도록 관리해주는 그룹웨어 시스템의 일종인 워크플로우 관리 시스템이 등장하게 되었다[1, 2].

EAI 이전의 애플리케이션 통합은 대개 직접적인 애플리케이션 코드의 수정 및 개발을 통해 연동시키는 일대일 방식(Point-to-Point)이 사용되었다. 하지만 많은 애플리케이션을 일일이 통합해야 하기 때문에 오랜 개발 기간과 고비용이 소요된다는 단점이 있으며 데이터와 애플리케이션이 변형될 때 이를 바꿔주는 유지보수비용도 상당한 문제였다. 이러한 단점을 극복하고자 미들웨어 기반의 어댑터를 사용하는 EAI가 등장하였지만, 이 역시 해당 미들웨어에 종속적이라는 문제를 여전히 가지고 있다[3].

본 논문은 애플리케이션의 변화를 최소화하면서 애플리케이션을 통합, 관리할 수 있는 미들웨어 독립적인 워크플로우 관리 시스템을 제안한다. 본 논문에서 제시하고 있는 워크플로우 관리 시스템은 애플리케이션 코드의 수정을 최소화하면서 미들웨어 독립적으로 애플리케이션을 통합할 수 있도록 이중 어댑터(Duplicate Adapter)를 사용하였고, 통합된 애플리케이션 사이의 원활한 데이터 교환을 위해서 XML을 사용하였다. 그리고 애플리케이션 사이의 상호작용 및 원활한 데이터의 흐름을 제어하는 워크플로우 모델링을 위하여 스테이트 차트와 스테이트 머신의 개념을 도입하였다.

본 논문의 구성은 다음과 같다. 우선 2장에서는 기반 연구로 어댑터 미들웨어 기반의 EAI와 워크플로우 관리 시스템에 대해서 소개하고 본 시스템 개발의 기반이 된 스테이트 차트 개념을 설명한다. 3장 및 4장에서는 제안하는 시스템의 분석 및 설계, 그리고 구현 및 평가에 대해서 논의한다. 마지막 5장에서는 결론 및 향후 연구 과제에 대해서 기술한다.

※ 본 연구는 2003년도 중앙대학교 교내 학술 연구비 지원에 의하여 연구되었음.
† 준 회 원 : 중앙대학교 대학원 컴퓨터공학과
†† 종신회원 : 중앙대학교 컴퓨터공학과 교수
논문접수 : 2003년 1월 27일, 심사완료 : 2003년 5월 19일

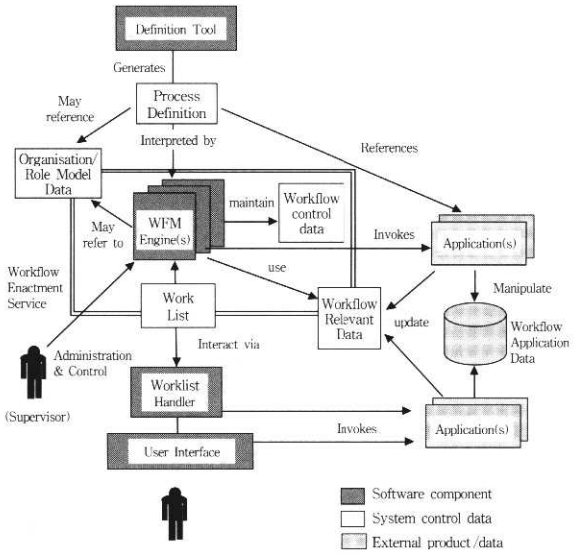
2. 기반 연구

2.1 EAI

EAI는 기업내부에 상이한 애플리케이션과 비즈니스 프로세스를 통합하는 IT 솔루션으로, 새로운 미들웨어를 이용해 기업 내 각종 애플리케이션을 통합하는 과정을 말한다[4]. 전통적인 미들웨어가 개별적인 애플리케이션을 일대일 방식으로 적용 했다면, EAI는 기업 내 상호 연관된 모든 애플리케이션을 유기적으로 연동시켜 필요한 정보를 통합하고 관리하고 사용할 수 있게 하는 솔루션이다. 이러한 EAI를 구축하기 위해서는 다음과 같은 3단계를 거치게 된다. 첫째, 애플리케이션간의 연결성을 메시지기반 미들웨어 혹은 애플리케이션 어댑터를 통해 연결성을 확보한다. 둘째, 이종의 데이터를 데이터브로커에 의해서 자료를 자동 변환한다. 셋째, 애플리케이션을 통합하는 비즈니스 프로세스를 자동화 도구 등을 통하여 생성한다[5]. 특히, 어댑터 미들웨어는 EAI 프로젝트에서 데이터 전달의 안정성과 성능을 보장하는 메시징 플랫폼으로, 서로 다른 포맷의 데이터를 자동으로 변환, 전달해주는 브로커, 비즈니스 프로세스를 자동화하는 워크플로우 엔진 등과 함께 애플리케이션 통합을 위한 필수 컴포넌트이다[6].

2.2 워크플로우 관리 시스템

워크플로우 관리 시스템은 크게는 비즈니스 프로세스의 자동화에서부터 작게는 애플리케이션 사이의 협력 작업을 관리하는 용도로 광범위하게 적용되고 있는 일종의 동적인 그룹웨어 시스템으로서, 기업 내외의 업무들과 관련된 사람들과 정보 및 기타 자원들의 흐름을 통합적으로 관리, 지원하기 위한 자동화된 도구들의 집합이다. 워크플로우 관리 시스템은 기업의 업무 흐름을 자동으로 처리함으로써 이에 따른 비용과 시간을 절감할 수 있는 솔루션으로 다양한 분야에 도입되어 생산성 향상 도구로 자리잡고 있다[7].

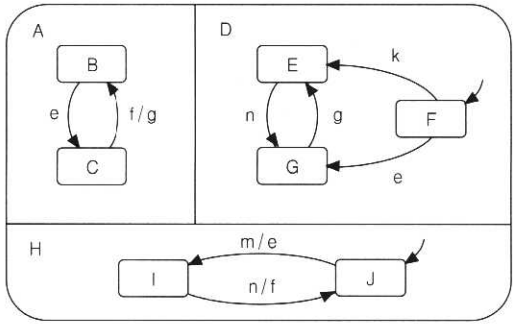


(그림 1) 워크플로우 관리 시스템의 일반 구조[8]

워크플로우 관리 시스템의 표준화를 진행하고 있는 표준단체인 WfMC(Workflow Management Coalition)는 워크플로우 관리 시스템 참조 모델(Workflow Management System Reference Model)[8]을 발표하여 워크플로우 관리 시스템의 표준적 모델을 제시하고 있다. (그림 1)은 WfMC의 워크플로우 참조 모델을 나타내고 있다.

2.3 스테이트 차트

스테이트 차트는 David Harel이 반응 시스템을 명세하기 위해 제한한 가시적 정형기법으로서[9, 10], 기존의 상태추이도(State transition diagram)에 계층성(hierarchy)과, 병행성(concurrency) 그리고 동보통신(broadcast communication)의 개념을 추가해 확장한 것으로서, 상태와 전이를 쉽게 클러스터화 하거나 구체화시킬 수 있는 정형 기법이다[11]. 계층성은 XOR 분할(decomposition)로 나타내어 질 수 있는데, 예를 들어 (그림 2)에서 상태 A는 B와 C의 XOR이 되며, 따라서 시스템이 상태 A에 있을 경우에는 상태 B나 상태 C중 하나에 존재하게 된다. 이에 반해 병행성은 두 개 이상의 XOR 분할이 AND 분할로 이루어져 있는 것을 말한다. 즉, (그림 2)에서 상태 A, D, H는 병행한 상태가 된다. 마지막으로 동보통신은 하나의 외부적 이벤트가 발생함으로 인해서 해당 이벤트와 관련된 모든 전이들이 발생될 수 있음을 나타낸다. (그림 2)에서 이벤트 f가 발생했을 때 f/g라는 레이블을 갖는 전이가 발생하면서 즉각적으로 이벤트 g에 대한 반응이 일어나게 된다. 만약 (그림 2)에서 시스템이 (B, F, J)에 있고 이벤트 m이 발생한다면, 다음 구성(configuration)은 이벤트 e에 의해서 트리거된 (C, G, I)가 될 것이다. 이를 길이 2의 연쇄반응(chain reaction)이라고 한다. 여기에서 이벤트 n이 발생하게 되면, 길이 3의 연쇄반응을 통해 새로운 구성 (B, E, J)가 될 것이다.



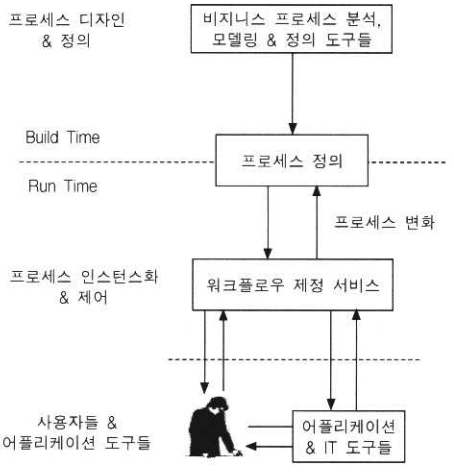
(그림 2) 스테이트 차트의 예

시스템의 동적인 동작을 기술하기 위한 정형 명세 언어로 사용하기 위하여 Harel의 스테이트 차트는 다음과 같이 보완될 수 있다. 각각의 상태 내에서는 트리거(trigger, 이

벤트 혹은 조건)가 정의되고, 트리거가 발생하면 전이(transition)가 일어날 목적지 상태(target state)와 이때 해 주어야 할 액션(action)이 정의된다. 그리고 현재 상태 내에서 실행해야 할 행위(activity)를 현재 상태에 처음 들어올 때 한번 행해주는 부분(entry action)과 상태에 머무르는 동안 행하는 부분(do action), 그리고 상태를 빠져나갈 때 해주어야 하는 부분(exit action)으로 나누어 정의한다[11].

3. 워크플로우 관리 시스템의 분석 및 설계

이 장에서는 (그림 3)의 WfMC의 워크플로우 관리 시스템 참조 모델[8]을 바탕으로 제안하는 워크플로우 관리 시스템의 주요 요구사항에 대해서 살펴보고, 그러한 요구사항을, 제안하고 있는 본 워크플로우 관리 시스템이 어떻게 만족 시키고 있는가에 대해서 설명하고자 한다.



(그림 3) WfMC 워크플로우 관리 시스템 참조 모델[8]

3.1 시스템 분석

기업 내의 애플리케이션들이 하나로 통합되기 위해서는 각각의 애플리케이션 정보가 올바로 모델링 되어야 하고, 그에 따른 적절한 워크플로우가 설계 되어야 한다. 그리고 적절히 모델링 된 워크플로우 데이터를 해석하고 실행할 수 있는 워크플로우 엔진이 필요하다. 기본적인 워크플로우 관리 시스템의 요구사항은 <표 1>과 같다.

〈표 1〉 워크플로우 관리 시스템 요구 사항

	요 구 사 항
1	애플리케이션에 포함되어 있는 서비스 정보의 모델링
2	워크플로우를 모델링 할 수 있는 기술 언어 정의
3	서비스를 호출할 수 있는 프로토콜이 정의
4	사용자 친화적 모델링 인터페이스를 제공
5	워크플로우를 모델링한 기술 언어 자료를 해석하고 실행 할 수 있는 엔진 제공
6	이 기종의 미들웨어 사이의 통신을 위한 어댑터를 제공

3.2 워크플로우 모델링

본 절에서는 스테이트 차트와 XML을 이용하여 애플리케이션 워크플로우를 모델링 하는 방법을 제시한다.

애플리케이션 사이의 워크플로우를 올바르게 모델링하기 위해서는 정적 모델링과 동적 모델링이 수행되어야 한다. 해당 애플리케이션이 가지고 있는 서비스의 목록과 이벤트를 비롯한 속성 정보 등을 모델링 하는 것이 정적 모델링이고, 이러한 정적 모델링 과정에서 정의된 애플리케이션 정보를 바탕으로 애플리케이션 사이의 워크플로우를 모델링 하는 것이 동적 모델링이다. 제안하는 워크플로우 관리 시스템에서는 이러한 정적, 동적 모델링을 위해서 스테이트 차트 기반의 그래픽 사용자 인터페이스 환경을 제공하고 있으며, 모델링 결과는 이 기종간 데이터 교환 표준인 XML로 저장된다. 제안하는 워크플로우 관리 시스템에서 제시하는 모델링 요소들은 SOAP(Simple Object Access Protocol)[12]의 경우 처럼 원격 프로시저 호출 기반의 통합환경에서 고려되어야 할 최소한의 요소를 중심으로 간단하게 설계되어 적용 환경에 따라 확장 가능하도록 하였다[13].

3.2.1 정적 모델링

정적 모델링은 애플리케이션 속성 정보를 모델링 하는 과정이다. 속성 정보에는 제공하는 서비스의 이름과 연결 타입(동기/비동기), 응답 제한 시간 및 사용되는 변수 등의 정보가 포함된다. 변수는 3.3절에서 소개하는 데이터 캐시에서 데이터 통합을 위해 사용되게 된다.

<표 2>는 ServiceA 1과 ServiceA 2를 가진 애플리케이션 A와 ServiceB를 가진 애플리케이션 B에 대한 정적인 속성 정보를 표현하고 있으며, 이들 정보에 대한 모델링 결과는 (그림 4)와 같은 XML로 표현된다.

〈표 2〉 애플리케이션의 서비스 정보

애플리케이션	서비스	연결타입	응답 제한 시간	변수
A	ServiceA 1	Sync	0 100 200	a b c
	ServiceA 2	Async	-	d
B	ServiceB	Sync	100	e

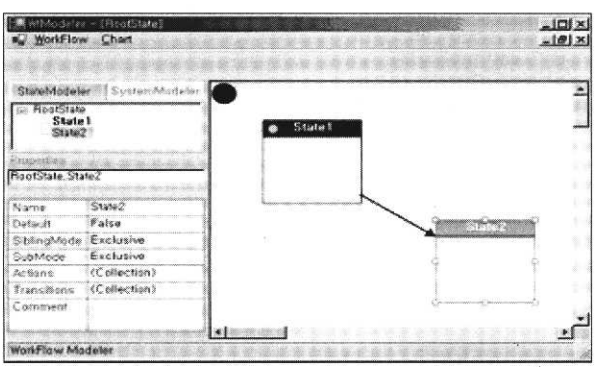
```
- < systems >
 - < system >
    < name > A < /name >
  - < services >
   - < service seqType = "S" >
      < name > ServiceA1 < /name >
    - < timeOuts >
       < timeOuts > 0 < /timeOuts >
       < timeOuts > 100 < /timeOuts >
       < timeOuts > 200 < /timeOuts >
      < /timeOuts >
    - < variables >
       < variable > a < /variable >
       < variable > b < /variable >
```

```
            < variable > c < /variable >
          < /variables >
        < /service >
      - < /service seqType = "A" >
          < name > ServiceA2 < /name >
          < timeOuts/ >
        - < variables >
            < variable > d < /variable >
          < /variables >
        < /service >
      < /services >
    < /system >
  - < system >
      < name > B < /name >
    - < services >
      - < service seqType = "S" >
          < name > ServiceB < /name >
        - < timeOuts >
            < timeOuts > 100 < /timeOuts >
          < /timeOuts >
        - < variables >
            < variable > e < /variable >
          < /variables >
        < /service >
      < /services >
    < /system >
  < /systems >
```

(그림 4) 애플리케이션 속성 정보에 대한 정적 모델링

3.2.2 동적 모델링

동적 모델링은 정적 모델링에서 생성된 정보들을 참조하여 애플리케이션간의 워크플로우를 기술하는 모델링을 말한다. 이러한 워크플로우는 기반 연구에서 소개한 David Harel의 스테이트 차트를 개선한 형태의 모델을 이용하여 정의된다. 메시지 혹은 이벤트 등에 의한 일련의 상태 변화를 스테이트로 기술하고, 각각의 스테이트에서 수행되어야 할 행위를 액션으로 정의함으로써 워크플로우의 동작을 모델링 할 수 있게 된다. 워크플로우의 동작은 스테이트 진입 시 수행할 행위와 스테이트에 머무는 동안 실행할 행위, 스테이트를 빠져 나갈 때 수행할 행위를 각각 스테이트 별로 작성한다. 각각의 스테이트는 이벤트나 메시지와 같은 트리거에 의해서 상태 전이가 이루어지며, 각각의 스테이트마다 정의되어진 액션에 따라서 워크플로우의 동적인 행위가 처리되게 된다.



(그림 5) 동적 모델링을 위한 사용자 인터페이스

```
< ?xml version = "1.0" encoding = "utf-8"? >
< stateDef >
  < state guid = "root" subMode = "EXC" siblingMode = "EXC"
        default = "N" >
    < name > RootState < /name >
    < transitioins >< /transitioins >
    < actions > < /actions >
    < state guid = "a" subMode = "EXC" siblingMode = "EXC"
          default = "Y" >
      < name > State1 < /name >
      < transitioins >
        < transitioin >
          < nextState > State2 < /nextState >
          < triggerDef >
            < trigger relation = "OR" >
              < trigger relation = "NO" >
                < system > A < /system >
                < service > ServiceA2 < /service >
              < /trigger >
            < /trigger >
          < /triggerDef >
          < actions >
            < action when = "TRANSITION" seqType = "S"
                  timeOut = "0" >
              < system > A < /system >
              < service > ServiceA1 < /service >
              < variable > b < /variable >
            < /action >
          < /actions >
        < /transitioin >
      < /transitioins >
      < actions > < /actions >
    < /state >
    < state guid = "b" subMode = "EXC" siblingMode = "EXC"
          default = "N" >
      < name > State2 < /name >
      < transitioins >< /transitioins >
      < actions >
        < actior when = "ENTRY" seqType = "S" timeOut = "0" >
          < system > B < /system >
          < service > ServiceB < /service >
          < variable > e < variable >
        < /action >
      < /actions >
    < /state >
  < /state >
< /stateDef >
```

(그림 6) 워크플로우의 동적 모델링 결과

(그림 5)는 동적 모델링을 위한 사용자 인터페이스의 모습으로, 애플리케이션 A의 ServiceA 2 이벤트가 발생하였을 경우 애플리케이션 B의 ServiceB 메소드를 호출하는 워크플로우를 모델링하고 있다. 초기 상태인 State 1에서 애플리케이션 A의 ServiceA 2 이벤트가 발생하면 State 1에서 State 2로 상태 전이가 발생하며 State 2의 ENTRY 액션으로 지정된 애플리케이션 B의 ServiceB가 실행된다. (그림 6)은 워크플로우의 동적 모델링을 수행한 결과로 생성되는 XML 문서이다.

(그림 6)에서 state 엘리먼트의 subMode/siblingMode 애트리뷰트는 자식 스테이트와 형제 스테이트간의 병행성 여부를 나타내고, trigger 엘리먼트는 상태 전이가 발생하게 되는 조건을 논리식으로 나타낸다.

3.3 데이터 캐시를 이용한 XML 데이터 통합

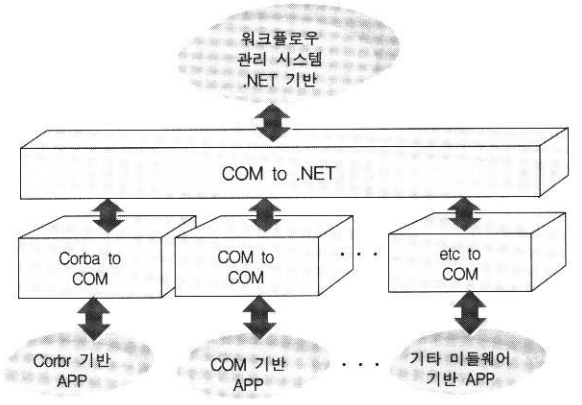
본 워크플로우 관리 시스템에서는 여러 애플리케이션들에 의해서 가공되는 데이터에 대한 보관과 전달을 위한 방법으로 워크플로우 데이터 레지스트리[14, 15]의 일종인 데이터 캐시를 사용하였다. 데이터 캐시는 워크플로우 관리 시스템 내부에 위치하게 되며, XML을 사용하여 데이터를 표현한다.

운용중인 애플리케이션들이 3.2에서 작성된 워크플로우 모델링 정보에 따라 작업을 수행하게 될 때에 워크플로우 관리 시스템은 애플리케이션들이 주고 받는 데이터를 공용 저장소인 데이터 캐시에 저장하여 관리한다. 데이터 캐시는 각각의 데이터에 대하여 데이터 내용과 데이터 값을 설정할 수 있으며, 라이프 사이클이 정의되어 있어서 생성 및 파기를 명확하게 표기할 수 있다.

3.4 이기종 미들웨어간의 통신

기존의 일반적 애플리케이션들은 서로 다른 미들웨어에 종속되어 운용되고 있다. 이러한 미들웨어가 서로 다른 환경에서 통신을 수행하기 위해서는 미들웨어에 대한 독립성이 확보되어야 한다.

본 워크플로우 관리 시스템에서는 통신상의 문제점을 해결하기 위해서 이중 어댑터를 사용하였다. 이중 어댑터 구조는 각 애플리케이션에서 사용하고 있는 미들웨어를 일반적으로 사용되어지고 있는 COM 연결로 변환하고 이를 다시 워크플로우 관리 시스템이 구현된 플랫폼의 미들웨어로 변환하는 이중 변환 과정을 거친다. (그림 7)은 본 워크플로우 관리 시스템의 COM 기반의 이중 어댑터 구조를 보여주고 있다.



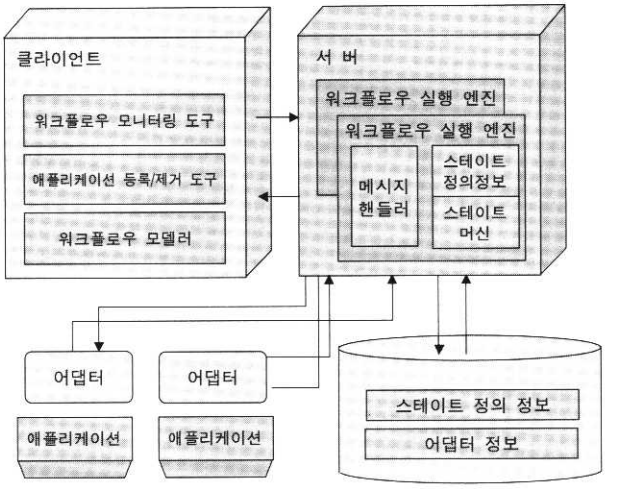
(그림 7) 미들웨어 독립성을 위한 이중 어댑터 구조

4. 워크플로우 관리 시스템의 구현 및 평가

4.1 워크플로우 관리 시스템의 구현

본 워크플로우 관리 시스템은 마이크로소프트의 닷넷 플랫폼에서 C#언어로 구현되었으며, XML과 스테이트 차트를 이용하여 설계하였고, 이기종 미들웨어 간의 메시징을 위해서 COM 기반의 이중 EAI 어댑터를 적용하여 구현하였다.

본 논문에서 제안하는 워크플로우 관리 시스템은 3-계층(3-tier) 모델을 사용하고 있다. 클라이언트 계층에서는 사용자가 쉽게 워크플로우를 모델링하고, 워크플로우의 동작을 모니터링 할 수 있는 사용자 친화적인 인터페이스를 제공한다. 그리고 데이터베이스 계층에서는 애플리케이션의 정적인 정보와 사용자가 모델링한 동적인 워크플로우 정보를 저장하며, 서버 계층에서는 데이터베이스 계층과 클라이언트 계층과 상호 작용하며 워크플로우를 관리하고 실행하는 워크플로우 엔진이 존재한다.



(그림 8) 워크플로우 관리 시스템 구조

워크플로우 엔진은 워크플로우를 해석하고 실행하는 핵심 요소이다. 워크플로우 엔진은 스테이트 정의 및 스테이트 머신으로 구성되어 있다. 메모리 상에 DOM 객체의 인스턴스로 관리되고 있는 워크플로우의 모델링 정보를 사용하여 스테이트 정의를 구성하게 되며, 스테이트 머신은 스테이트 정의에 따라 각 스테이트를 순회하면서 정의된 작업을 수행하게 된다. 그리고 스테이트 머신은 스테이트 정의에 정의된 트리거 정보에 해당하는 이벤트나 메시지가 감지되면 다음 스테이트로 전이된다. 이러한 전이와 수행을 반복하면서 모든 워크플로우 작업을 실행하게 된다.

4.2 비교 및 평가

일반적인 워크플로우 관리 시스템들에 비해 본 워크플로우 관리 시스템이 지닌 특징은 스테이트 차트 개념의 적용, 이중 어댑터의 사용, XML 기반의 데이터 캐시의 도입으로 요약할 수 있다.

본 논문에서 제안하는 워크플로우 관리 시스템은 스테이트 차트에 기반을 두고 있기 때문에 워크플로우 디자이너에게 스테이트 차트 작성 능력을 요구하게 되지만, 스테이트 머신으로 구현된 워크플로우 엔진에서 해석이 용이하여

효율적 워크플로우의 운용이 가능하게 된다는 장점이 있다. 그리고 이중 어댑터의 적용은 기존 애플리케이션과 워크플로우 관리 시스템의 미들웨어 종속성을 해결할 수 있는 간단하면서도 효과적 방안이라 할 수 있다. 또한 XML 기반의 워크플로우 데이터 레지스트리 개념을 도입한 데이터 캐시의 사용 또한 데이터 관리 및 전송의 효율성을 높이는 동시에 호환성을 유지하게 하는 중요한 요소로 작용하고 있다.

5. 결론 및 향후 연구 과제

제안하고 있는 워크플로우 매지니먼트 시스템은 이중 어댑터를 사용하여 기존의 애플리케이션의 수정을 최소화하고도 애플리케이션 통합을 가능하게 하였으며 스테이트 차트의 개념을 도입해 효율적인 워크플로우 모델링을 가능하게 하였다.

하지만 제안한 워크플로우 관리 시스템의 워크플로우 엔진은 기존 애플리케이션에서 올라온 데이터를 런타임에 동적으로 가공하지 못하는 한계를 지니고 있다. 향후에는 어댑터 레벨에서의 데이터 가공 뿐만이 아니라 표준 스크립트 언어를 이용하여 런타임에 데이터를 가공함으로써 더욱 더 유연한 애플리케이션 통합 환경을 제공할 수 있는 방안이 연구되어야 한다. 그리고 워크플로우 엔진에 부하가 집중되어 발생하는 병목현상을 효율적으로 해결할 수 있는 방안 또한 고려되어야 한다. 마지막으로, 일반적인 워크플로우 관리 시스템과 스테이트 차트를 도입한 본 시스템 사이의 성능을 구체적인 수치 데이터로 비교 분석할 수 있는 합리적 실험을 통한 평가 및 검증 과정이 요구된다.

참 고 문 헌

[1] 한현철, 전자연, 윤철, "비즈니스 프로세스 중심의 XML 기반 통합 브로커 설계 및 구현", 정보과학회 2001년 추계학술대회, Vol.28, No.02, pp.0712-0714, 2001.

[2] 한동수, 심재용, "워크플로우 및 워크플로우 관리 시스템의 새로운 조망", 한국정보과학회논문지D, Vol.28, No.03, pp. 395-405, 2001.

[3] EAI Journal, "http://www.eaijournal.com."

[4] Francis X. Mainnis, William A. Ruh, "Enterprise Application Integration," John Wilcy & Sons, 2000.

[5] 정성혜, 이은서, 이경환, "EAI를 위한 통합 메시지 어댑터", 정보과학회 2001년 추계학술대회, Vol.28, No.02, pp.478-480, 2001.

[6] 전자신문, "http://etimesi.com", 2002.

[7] 신동일, 신동규, "워크플로우 관리 시스템의 설계 및 구현", 정보처리학회논문지, 제7권 제5호, 2000.

[8] David Hollingsworth, "The Workflow Reference Model," Workflow Management Coalition, 1995.

[9] D. Harel, "Statecharts : A Visual Formalism for Complex Systems," Science of Computer Programming, 8, pp.231-274, 1987.

[10] D. Harel, A. Pnueli, J. P. Schmidt and R. Sherman, "On the formal semantics of statecharts," Proc. 2nd IEEE Sysposium on Logic in Computer Science, pp.54-64, 1987.

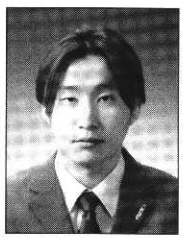
[11] 김철웅, 한상용, 최진영, 이정아, "IP/Sim : Statecharts에 기반을 둔 가상 프로토타이핑 시뮬레이터 설계 및 구현", 정보처리학회논문지, 제7권 제3호, pp.891-900, 2000.

[12] Simple Object Access Protocol(SOAP), "http://www.w3.org/2000/xp/Group/#soap12," 2002.

[13] 송종만, 이선현, 문기동, 김광훈, 백기수, "XML 기반 워크플로우 응용 프로그램 호출 매커니즘", Vol.08, No.01, pp.47-50, 2001.

[14] 김중일, 이이섭, 백두권, 나홍석, "워크플로우 시스템 사이의 상호운영성을 위한 데이터 교환 프레임워크", 한국정보과학회논문지B, Vol.28, No.03, pp.249-261.

[15] 원재강, 김학성, 김광훈, 정관희, "워크플로우를 위한 저장소 관리 시스템," 정보처리학회 춘계학술대회, Vol.08, No.01, pp.97-100, 2001.



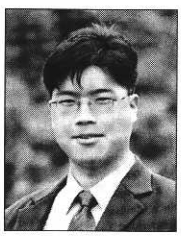
윤 석 현

e-mail : lazecool@archi.csc.cau.ac.kr

2002년 중앙대학교 공과대학(공학사)

2002년~현재 중앙대학교 대학원 컴퓨터공학과 석사과정

관심분야 : 웹 서비스, ebXML, 시맨틱웹



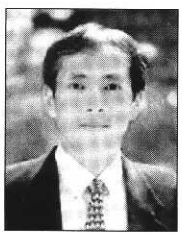
오 명 은

e-mail : jmania@archi.csc.cau.ac.kr

2002년 중앙대학교 공과대학(공학사)

2002년~현재 중앙대학교 대학원 컴퓨터공학과 석사과정

관심분야 : EAI, 비즈니스 워크플로우, 웹 서비스



한 상 용

e-mail : hansy@cau.ac.kr

1975년 서울대학교 공과대학(공학사)

1984년 Minnesota 공과대학(공학박사)

1984년~1995년 IBM 책임연구원

1995년~현재 중앙대학교 컴퓨터공학과 교수

관심분야 : Virtual Prototyping, EC(Electronic Commerce), Internet Application